VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA,BELAGAVI – 590018

KARNATAKA

Mini Project Report
On
"Solving Travelling Salesman Problem Using Greedy Algorithm and BruteForce Algorithm"

SUBMITTED IN PARTIAL FULFILLMENT OF THE ASSIGNMENT

FOR THE Analysis & Design of Algorithms (BCS401)
COURSE OF IV SEMESTER

Submitted by

Name: MADIHA KHANUM

USN: 1CG23CS403

Guide:
Mr.Asif Ulla Khan M. Tech.
Asst. Prof., Dept. of CSE
CIT, Gubbi.

HOD:
Dr. Shantala C P PhD.,
Head, Dept. of CSE
CIT, Gubbi.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(NAAC Accredited & ISO 9001:2015 Certified Institution)
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka

2023-24

Rubric – B.E. Mini-Project [BCS401]

Course outcome	Rubric/Level	Excellent (91-100%)	Good (81-90%)	Average (61-80%)	Moderate (40-60%)	Score
CO1	Identification of project proposal (05 Marks)					
CO2	Design and Implementation (10 Marks)					
CO3	Presentation skill (05 Marks)					
CO4	Individual or in a team development					
CO5	Report (05 Marks)					
Total						

Course outcome:

CO 1: Identification of project proposal which is relevant to subject of engineering.

CO 2: Design and implement proposed project methodology.

CO 3: Effective communication skill to assimilate their project work.

CO 4: Work as an individual or in a team in development of project.

C0 5: Understanding overall project progress and performance.

Student Signature **Faculty signature**

ABSTRACT

Greedy algorithm is an algorithm that will solve problem by choosing the best choice/optimum solution at that time, without considering the consequences that will affect it later, thus the solution won't be always the optimal global solution, while Brute force algorithm is an algorithm that is using straightforward method to solve a problem.

There are several algorithms that are naturally a Greedy algorithm that can be used to solve this problem, which are the Prim's Minimum Spanning Tree, Dijkstra shortest path, Huffman coding, and Kruskal Minimum Spanning Tree, while for brute force can be used in almost every problems. In this paper, we will use brute force and greedy algorithm to solve Travelling Salesman Problem. Travelling Salesman Problem (TSP) is one of the NP-Complete problem that will search for optimalsolution when a salesman should precisely visit every city once and then back again to the initial city.

The main aim of this problem is to find the shortest possible route for the salesman to visit. The presented paper will use brute force algorithm and greedy algorithm to solve travelling salesman problem and examine those two algorithms' time complexity, also comparing them. The result is greedy algorithm is more efficient than brute force algorithm, but the consequences is that greedy algorithm won't always give the optimal solution.

The Travelling Salesman Problem (TSP) is a classic optimization problem in combinatorial optimization, where the objective is to find the shortest possible route that visits a set of cities exactly once and returns to the origin city. TSP has significant applications in logistics, planning, and manufacturing.

Two well-known approaches to solving the TSP are the greedy algorithm and the brute force algorithm. The greedy algorithm constructs a solution iteratively, choosing the next city based on a locally optimal criterion, such as the shortest distance. This approach is computationally efficient and easy to implement but often yields suboptimal solutions since it does not consider the global structure of the problem.

The brute force algorithm exhaustively searches all possible routes to guarantee the optimal solution. By evaluating every permutation of cities, the brute force method ensures finding the shortest path. However, this approach is computationally infeasible for large instances due to its factorial time complexity.

CHAPTER 1:

INTRODUCTION

In this final project, we choose “Solving Travelling Salesman Problem using Greedy Algorithm and Brute Force Algorithm”.

In this paper, we will discuss about Travelling Salesman Problem (TSP) using methods. The Travelling Salesman Problem or the TSP is a representative of a large class of problems known as combinatorial optimization problems (Mohammad Reza Bonyadi, 2008). In the ordinary form, it is represented with a salesman and cities. Salesman has to visit each cities, which is built on one another and has different distances, one by one, starting from his hometown and returning to his hometown again. The aim for this problem is to find the shortest possible route to minimize the length of the trip.

Brute Force is a type of algorithm that tries a large number of pattern to solve a problem (Spacey, 2016). This means that in TSP, the algorithm will try all possible routes and compute their distance to find the minimum distance. Therefore, this algorithm will always give optimal solution.

Greedy algorithm is an algorithm that will solve problem by choosing the bestchoice/optimum solution at that time, without considering the consequences that will affect it later. In many problems, a greedy strategy does not in general produce an optimal solution, but nonetheless a greedy heuristic may yield locally optimal solutions that approximate a global optimal solution in a reasonable time (Ejim, 2016).

In this paper, we will try to solve TSP with Brute Force and Greedy Algorithm, and we will compare them and see which one is better for solving this problem.

The Travelling Salesman Problem (TSP) is a fundamental problem in combinatorial optimization and computer science, defined as finding the shortest possible route that visits each city in a given set exactly once and returns to the starting point. This problem is NP-hard, meaning that no polynomial-time algorithm is known to solve all instances of TSP optimally. TSP has numerous practical applications, including logistics, route planning, manufacturing, and DNA sequencing.

CHAPTER 2:

PROBLEM STATEMENT

Brute Force Algorithm

The TSP solutions for brute force is by trying all possible routes also compute the total distance and then compare the distances from each routes. The least distance will be taken as the optimal solution. The following are the functions that we use in our program.

- Def takeInput(n)
- Def permutation(lst)
- Def countCost(matrix)

Greedy Algorithm

For solving TSP, the algorithm will selects the shortest distance for each initial city to target city. After adding up the distance from each city, the algorithm choose the shortest possible distance and also the routes.

- Def takeInput(n)
- Def countCost(matrix)

Time Complexity

TSP solutions in the algorithm imply trying all the permutation combinations with a complexityof O(n! (n The + 1)). with n is the number of cities. If we want to solve 4 city problem, the time running would be 120. If we try with 8 cities, the time would be 362880, we can see that this algorithm won't be practical to use with problem that has many cities. The complexity of TSP using greedy algorithm O($n^2 + 2n + 2$) with n is the number of cities on the problem. If we want to solve 4 city problem, the time running would be 26, andif we solve 8 city problem, the time running would be 82.

CHAPTER 3:

IMPLEMENTATION

TSP with Greedy Program Code

```
import time
import math
matrix = []
completed = []
numCity = 0
def takeInput(numCity):
matrix = [[0 for j in range(numCity)]
for i in range(numCity)]
for x in range (0, numCity):
print("")
print("Enter Elements of Row: ",x)
for y in range(0, numCity):
print("Column",y,": ",end ="")
matrix[x][y] = int(input())
completed.append(0)
print("")
print("The distance list is:")
print(" ", end =" ")
for c in range (0, numCity):
print(c , end =" ")
print("")
print(" ", end =" ")
for d in range (0, numCity):
print("-" , end =" ")
print("")
for a in range (0, numCity):
print(a, "|", end =" ")
for b in range(0, numCity):
print(matrix[a][b], end =" ")
print("")
```

```
        return matrix
        def countCost(numCity,matrix):
        a = 0
        col = 0
        nextCity      =      0
        print("") print("Best
        route: ") while a !=
        numCity:
        minDist = 9999
        for b in range (0, numCity):
        if (matrix[col][b] < minDist) & (matrix[col][b] != 0) &(completed[b] != 1):
minDist = matrix[col][b]nextCity = b
completed[col] = 1 print(col,"--> ",end ="")col =
nextCity
                a = a + 1
                print("0")
          print("TSP WITH GREEDY ALGORITHM")
          print("-----------------------------------------------")
          numCity = int(input("Enter the number of cities: "))matrix =
          takeInput(numCity)
          start_time = time.perf_counter()
          countCost(numCity, matrix)
          print('solved in', time.perf_counter() - start_time, 'seconds')
```

TSP with Brute Force program code

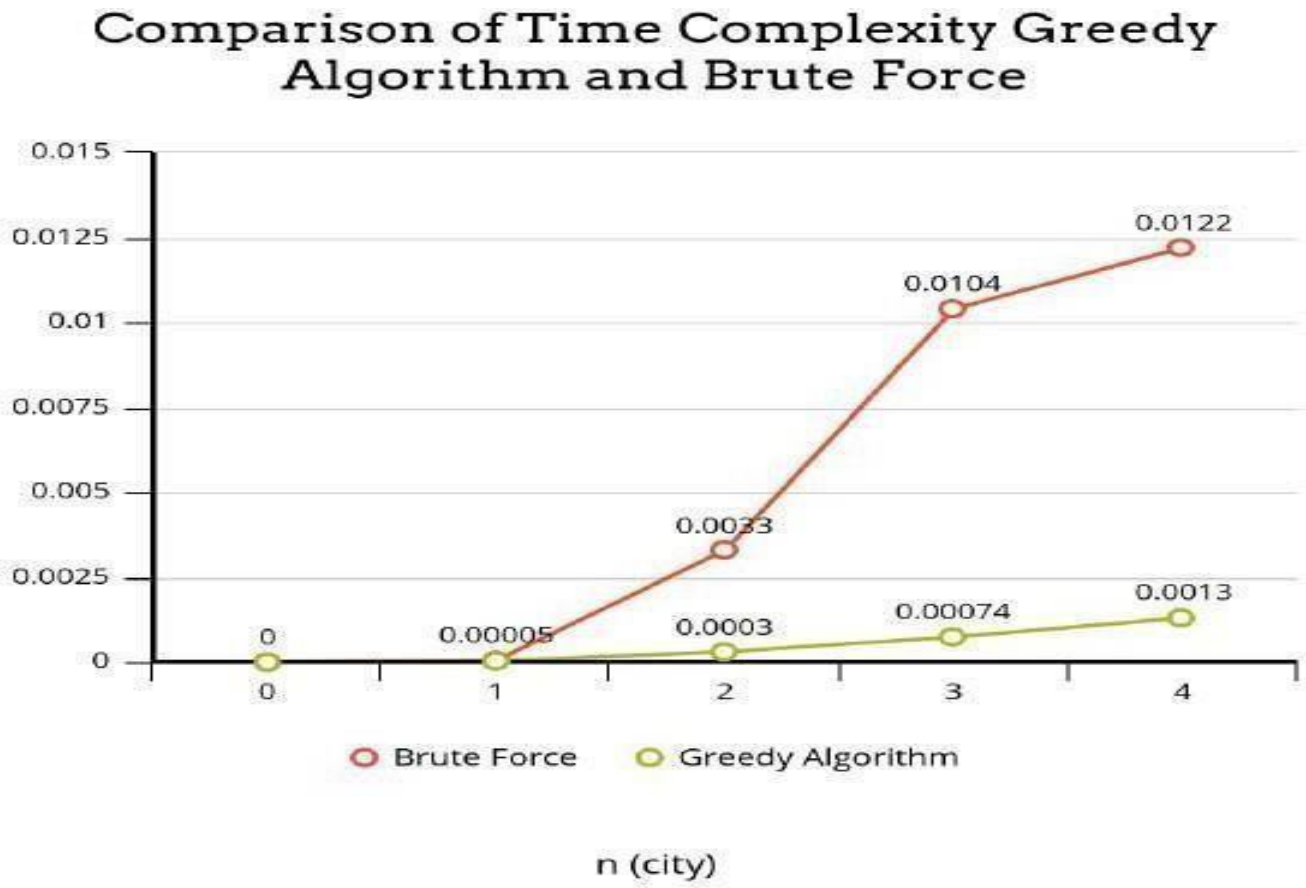
```
import time
import math
from itertools
 import permutations
matrix= []
comp=[]
numCit=0
def takeInput(numCity):
matrix = [[0 for j in range(numCity)]
for i in range(numCity)]
for x in range (0, numCity):
        print("")
print("Enter Elements of Row: ",x)
for y in range(0, numCity):
print("Column",y,": ",end ="")
matrix[x][y] = int(input())
completed.append(0)
print("")
print("----------")
print("")
print("The distance matrix is:")
print("  ", end =" ")
for c in range (0, numCity):
    print(c , end =" ")
    print("")
    print("  ", end =" ") for
d in range (0, numCity):
print("-" , end =" ")
    print("")
    for a in range (0, numCity):
    print(a, "|", end =" ")
    for b in range(0, numCity):
        print(matrix[a][b], end =" ")
            print("")
```

```
 return matrix
def permutation(lst):
 if len(lst) == 0:
 return []
 if len(lst) == 1:
 return [lst]
 l = []
 for i in range(len(lst)):
 m = lst[i]
 remLst = lst[:i] + lst[i+1:]
 for p in permutation(remLst):
 l.append([m] + p)
 return l
 def countCost(matrix):
 l = list(permutations(range(1, numCity)))
 print("")
 print("      ")
 print("")
 print ('list of possible routes (without starting point and end point): ',l)
 print("")
 i = 0
  min = 9999999
 minRoute= []
 while i<len(l):
 print(0,"--> ",end ="")
 dist = matrix[0][l[i][0]] j = 0
 while j<numCity-1:
 if (j == numCity-2):
 print(l[i][j], end ="")
 else:
 dist = matrix[l[i][j]][l[i][j+1]] + dist
 print(l[i][j],"--> ",end ="")
  j = j + 1
```

```
        dist = matrix[0][l[i][j-1]] + dist if (dist < min):
        min = dist  minRoute = l[i]
        print("---> 0")
        print(" distance:",dist)
        print("")
        i = i + 1
       return minRoute
print("TSP WITH BRUTE FORCE")
print("------------------------------------------")
numCity = int(input("Enter the number of cities: "))
matrix = takeInput(numCity) start_time = time.perf_counter()
minRoute = countCost(matrix)
print("------------------------------------------------------------------")
print("")
print('Best route: ', 0,"-->",end =" ")
print(*minRoute, sep =" --> ",end =" ")
print("-->", 0)
print('solved in', time.perf_counter() - start_time, 'seconds')
```

CHAPTER 4:

RESULTS



The image below illustrates the TSP problem using 4 cities.

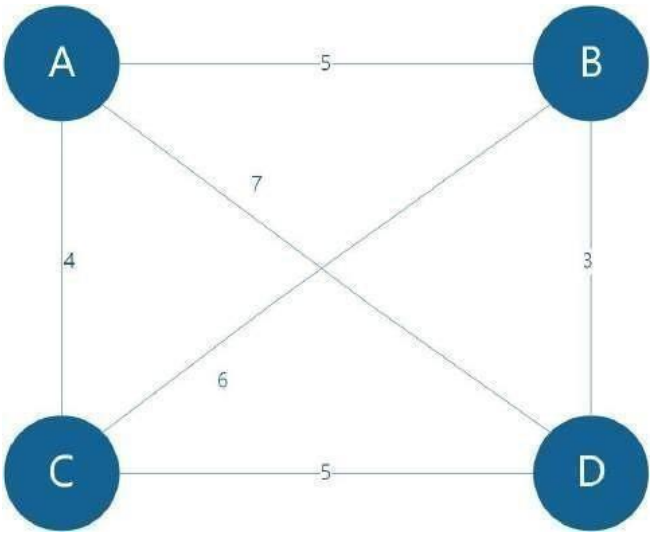


Figure 1 - Graph of Travelling Salesman Problem with 4 Cities.

The image below illustrates the TSP problem using 3 cities.

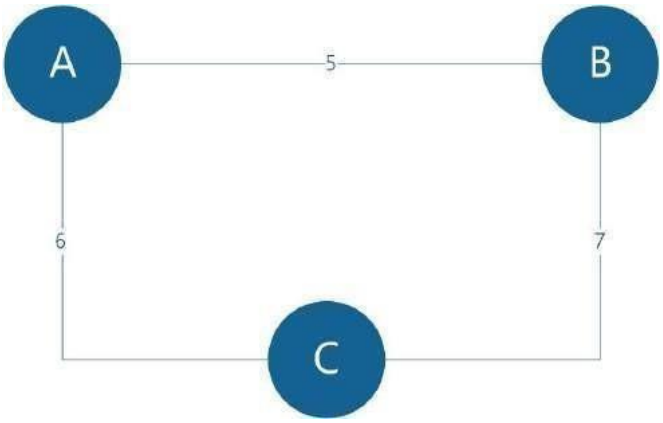


Figure 2 - Graph of Travelling Salesman Problem with 3 Cities.

The image below illustrates the TSP problem using 2 cities.

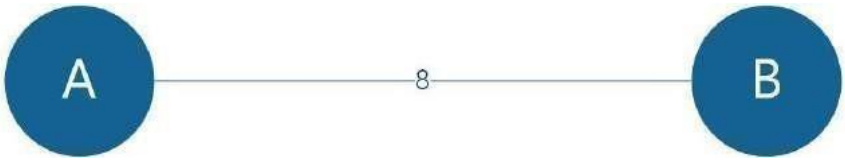


Figure 3 - Graph of Travelling Salesman Problem with 2 Cities

CHAPTER 5:

CONCLUSION

According to our experiment, solving TSP with brute force algorithm and greedy algorithm, we can conclude that greedy algorithm is more efficient than brute force algorithm, because the running time for greedy algorithm is much faster than brute force algorithm. But the consequences is that greedy algorithm will not always gives the optimal solution, while brute force will take much time, but alwaysgives the optimal solution.

REFERENCES:

[1] Ejim, S. (2016, September). Implementation of Greedy Algorithm in Travel Salesman Problem. Retrieved from ResearchGate: https://www.researchgate.net/publication/307856959_Implementation_of_Greedy_Alogrithm in_Travel_Salesman_Problem.

[2] Mohammad Reza Bonyadi, M. R. (2008, September 1). Population-Based Optimization Algorithms for Solving the Travelling Salesman Problem. Retrieved from IntechOpen: https://www.intechopen.com/books/traveling_salesman_problem/population-based_optimization_algorithms_for_solving_the_travelling_salesman_problem.

[3] Spacey, J. (2016, March 29). What is a Brute Force Algorithm? Retrieved from Simplicable: https://simplicable.com/new/brute-force